SendKeys Class

Namespace: System.Windows.Forms

Assembly: System.Windows.Forms.dll

Provides methods for sending keystrokes to an application.

In this article

C# Copy

```
public class SendKeys
```

Inheritance Object → SendKeys

Examples

The following code example demonstrates how to use the Send method. To run the example, paste the following code in a form called Form1 containing a button called Button1. Ensure the click events are associated with their event-handling methods in this example. The button control's TabIndex property should be set to 0. When the example is running, double-click the form to trigger the button's click event.

C# Copy

```
// Clicking Button1 causes a message box to appear.
private void Button1_Click(System.Object sender, System.EventArgs e)
{
    MessageBox.Show("Click here!");
}
```

```
// Use the SendKeys.Send method to raise the Button1 click event
// and display the message box.
private void Form1_DoubleClick(object sender, System.EventArgs e)
{

    // Send the enter key; since the tab stop of Button1 is 0, this
    // will trigger the click event.
    SendKeys.Send("{ENTER}");
}
```

Remarks

Use SendKeys to send keystrokes and keystroke combinations to the active application. This class cannot be instantiated. To send a keystroke to a class and immediately continue with the flow of your program, use Send. To wait for any processes started by the keystroke, use SendWait.

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, pass in the string "A" to the method. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, specify the parameter as "ABC".

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses () have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use "{+}". To specify brace characters, use "{{}" and "{}}". Brackets ([ ]) have no special meaning to SendKeys, but you must enclose them in braces. In other applications, brackets do have a special meaning that might be significant when dynamic data exchange (DDE) occurs.

> **⊗ Caution**
>
> If your application is intended for international use with a variety of keyboards, the use of **Send** could yield unpredictable results and should be avoided.

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes in the following table.

Key	Code
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC} (reserved for future use)
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}

Key	Code
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}
Keypad add	{ADD}
Keypad subtract	{SUBTRACT}
Keypad multiply	{MULTIPLY}
Keypad divide	{DIVIDE}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes.

Key	Code
SHIFT	+
CTRL	^
ALT	%

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 42} means press the LEFT ARROW key 42 times; {h 10} means press H 10 times.

> **Note**
>
> Because there is no managed method to activate another application, you can either use this class within the current application or use native Windows methods, such as `FindWindow` and `SetForegroundWindow`, to force focus on other applications.

> **Note**
>
> The SendKeys class has been updated for the .NET Framework 3.0 to enable its use in applications that run on Windows Vista. The enhanced security of Windows Vista (known as User Account Control or UAC) prevents the previous implementation from working as expected.
>
> The SendKeys class is susceptible to timing issues, which some developers have had to work around. The updated implementation is still susceptible to timing issues, but is slightly faster and may require changes to the workarounds. The SendKeys class tries to use the previous implementation first, and if that fails, uses the new implementation. As a result, the SendKeys class may behave differently on different operating systems. Additionally, when the SendKeys class uses the new

implementation, the SendWait method will not wait for messages to be processed when they are sent to another process.

If your application relies on consistent behavior regardless of the operating system, you can force the SendKeys class to use the new implementation by adding the following application setting to your app.config file.

```
<appSettings>

<add key="SendKeys" value="SendInput"/>

</appSettings>
```

To force the SendKeys class to use the previous implementation, use the value `"JournalHook"` instead.

Methods

Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
Flush()	Processes all the Windows messages currently in the message queue.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetType()	Gets the Type of the current instance. (Inherited from Object)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
Send(String)	Sends keystrokes to the active application.
SendWait(String)	Sends the given keys to the active application, and then waits for the messages to be processed.
ToString()	Returns a string that represents the current object. (Inherited from Object)

Applies to

.NET Core

3.0 Preview 8

.NET Framework

4.8, 4.7.2, 4.7.1, 4.7, 4.6.2, 4.6.1, 4.6, 4.5.2, 4.5.1, 4.5, 4.0, 3.5, 3.0, 2.0, 1.1
